RELIABLE FILE TRANSFER IN GRID ENVIRONMENTS

Ravi K Madduri
Argonne National Laboratory, Illinois Institute of Technology
madduri@mcs.anl.gov
Cynthia S. Hood
Illinois Institute of Technology
hood@iit.edu
William E. Allcock
Argonne National Laboratory
allcock@mcs.anl.gov

Abstract

Grid-based computing environments are becoming increasingly popular for scientific computing. One of the key issues for scientific computing is the efficient transfer of large amounts of data across the Grid. In this poster we present a Reliable File Transfer (RFT) service that significantly improves the efficiency of large-scale file transfer. RFT can detect a variety of failures and restart the file transfer from the point of failure. It also has capabilities for improving transfer performance through TCP tuning.

1 Introduction

As scientific computing efforts make increasing use of Grid-based computing environments, file transfer operations become a key focus. Data-intensive, high-performance computing applications require the efficient management and transfer of terabytes or petabytes of information across the Grid. Examples of such applications include experimental analyses and simulations in scientific disciplines such as high-energy physics, climate modeling, earthquake engineering, and astronomy. In these applications, massive datasets must be transferred between the machines involved in the computation. Although the underlying file transfer utilities currently in use are well established and continue to be improved in terms of both efficiency and security, they do not provide a persistent file transfer service that can be easily invoked and trusted to complete in all but the most dire of circumstances. In this paper we describe a Reliable File Transfer (RFT) service that significantly improves the efficiency of large-scale file transfer.

The Reliable File Transfer Service is built on top of the existing GridFTP [1] client libraries. It inherits the performance and features provided by GridFTP such as restart support. However, GridFTP requires that the client remains active until the transfer finishes. Loss of the client state machine requires a manual restart from scratch. This situation motivates the development of Reliable File Transfer Service, a non-user-based service where a user can submit a request for transferring a set of files and free his/her local desktop or laptop. The transfer state is stored in a persistent manner so that in case of failure, the transfer is started not from scratch but from the last restart marker recorded for that transfer. The architecture of RFT is shown in Figure 1.

We employ a proactive approach to detect failures and recover gracefully. Specifically, we evaluate the information available at various levels, match the actions that can be taken with the information that is available, and take action appropriately at the application level. The information that is available at various levels is described below.

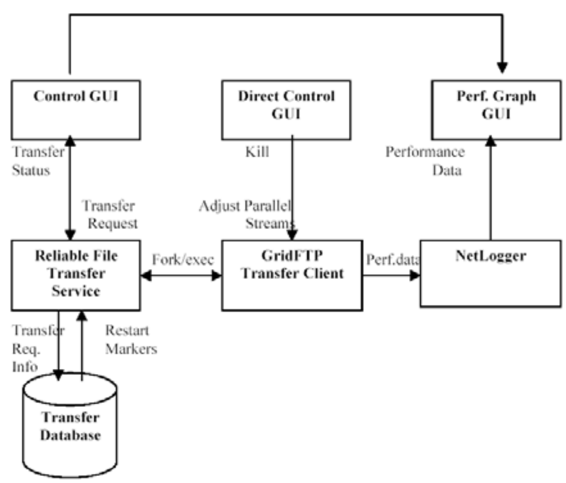

Figure 1. RFT architecture

1.1 Application Level

At the application level the file transfer client program performs the transfer. The client can be monitored by having a thread wait on it. Using the exit value process, this thread detects when the program crashes and restarts the process if the exit value indicates that the process has crashed or failed. The state of the application may be stored to a persistent storage so that it can be restored.

1.2 Network Level

At the network level we receive information from TCP. TCP is a connection-oriented protocol that keeps track of the packets sent from the sender and checks to see whether these packets are received at the receiver end. TCP has information about the packets that are not acknowledged. The TCP retry mechanism takes care of packet losses and retransmissions; but when there is a network outage, the connection cannot be saved through TCP. The client that is performing the transfer uses acknowledgment timeouts to detect that the host that it is trying to transfer to is no longer reachable. The destination host can become unreachable for many different reasons. At this point the client must wait for the network problems to be resolved and then retry the transfer. The client will retry the transfer periodically until it either goes through or times out.

The TCP buffer size can be adjusted by using negotiation between the source and destination machines. This negotiation can be automated and has the potential to increase the efficiency of the data transfer. Data can also be transferred in multiple streams, thereby reducing the impact of network congestion and subsequent setin of the TCP back-off mechanism on any one of the streams.

1.3 System Level

For a file transfer operation, a system-level failure can occur in three different places that are involved in the data transfer: the source host, the destination host, and the host that controls the transfer between these two hosts. If a particular host crashes or gets rebooted by the system administrator, the application can recover from the crash if its state has been stored to a secondary storage device. Recovery can be achieved by having the application start along with other system services and recover its state from disk.

2 Implementation

RFT monitors the state of the transfer and recovers from a variety of failures. The Reliable File Transfer service performs two services:

- It implements the transfer restart mechanisms provided for by GridFTP [1]. These mechanisms handle the failure of the RFT service itself or of either of the GridFTP data mover processes.
- It allows for the failure and restart of client applications. Upon restart, the applications can reconnect to the data transfer requests that they previously initiated. The data transfer requests are stored to persistent secondary storage so they can be recovered after a failure.

The Failure Recovery Mechanism provides RFT with the ability to recover from failures, such as server crashes and network outages. The Java service forks off the transfer client with appropriate parameters and then monitors the transfer by waiting on the transfer client. The transfer client is programmed to provide the service three kinds of exit codes. The success code is returned when the transfer client successfully transfers a given file. The fatal code is returned when the transfer cannot be completed; it indicates that there is a problem that cannot be solved with a simple retry mechanism. The nonfatal error code encompasses anything that does not fit into the other two exit codes. It denotes that something went wrong while trying to perform that transfer, but the transfer can be recovered from by a simple retry.

When the client returns a fatal error (e.g., when the source URL or destination URLs are not valid), the service will not restart the failed transfer. When the client returns a nonfatal error (which can be anything from a crashed server to network outage), the service will restart the transfer from the last restart marker received. The number of times the service tries to restart a transfer can be configured before starting the service. RFT provides failure recovery mechanisms for all the failure conditions by taking predetermined appropriate action depending on the information received at different levels.

3 Summary

RFT was thoroughly tested and found to recover from a variety of failures. The longest test ran for three days, transferring 0.3 terabytes of data. It was showcased at Supercomputing 2001 as one of the applications from Argonne National Laboratory. Future work includes automatic TCP tuning and adjustment of the number of parallel streams. RFT is now one of the higher-level services in the Globus Toolkit™. We are also working on integrating RFT with the Replica Catalog Selection Service of the Globus Toolkit™.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-ENG-38.

Reference

[1] GridFTP: Universal Data Transfer for the Grid
http://www.globus.org/datagrid/gridftp.html